SIEMENS

COMOS

Platform Programmation

Manuel d'utilisation

Marques 1

Composants .NET pour COMOS 2

Développement avec Visual Basic 6 dans COMOS 3

Configuration Side-by-Side à l'aide de manifestes 4

WinCC 5

ControlProperties 6

Créer et intégrer les symboles 3D 7

Spécifications d'icônes 8

Autres composants 9

09/2011

A5E03638268-01

Mentions légales

Signalétique d'avertissement

Ce manuel donne des consignes que vous devez respecter pour votre propre sécurité et pour éviter des dommages matériels. Les avertissements servant à votre sécurité personnelle sont accompagnés d'un triangle de danger, les avertissements concernant uniquement des dommages matériels sont dépourvus de ce triangle. Les avertissements sont représentés ci-après par ordre décroissant de niveau de risque.

⚠ DANGER
signifie que la non-application des mesures de sécurité appropriées **entraîne** la mort ou des blessures graves.

⚠ ATTENTION
signifie que la non-application des mesures de sécurité appropriées **peut entraîner** la mort ou des blessures graves.

⚠ PRUDENCE
accompagné d'un triangle de danger, signifie que la non-application des mesures de sécurité appropriées peut entraîner des blessures légères.

PRUDENCE
non accompagné d'un triangle de danger, signifie que la non-application des mesures de sécurité appropriées peut entraîner un dommage matériel.

IMPORTANT
signifie que le non-respect de l'avertissement correspondant peut entraîner l'apparition d'un événement ou d'un état indésirable.

En présence de plusieurs niveaux de risque, c'est toujours l'avertissement correspondant au niveau le plus élevé qui est reproduit. Si un avertissement avec triangle de danger prévient des risques de dommages corporels, le même avertissement peut aussi contenir un avis de mise en garde contre des dommages matériels.

Personnes qualifiées

L'appareil/le système décrit dans cette documentation ne doit être manipulé que par du **personnel qualifié** pour chaque tâche spécifique. La documentation relative à cette tâche doit être observée, en particulier les consignes de sécurité et avertissements. Les personnes qualifiées sont, en raison de leur formation et de leur expérience, en mesure de reconnaître les risques liés au maniement de ce produit / système et de les éviter.

Utilisation des produits Siemens conforme à leur destination

Tenez compte des points suivants:

⚠ ATTENTION
Les produits Siemens ne doivent être utilisés que pour les cas d'application prévus dans le catalogue et dans la documentation technique correspondante. S'ils sont utilisés en liaison avec des produits et composants d'autres marques, ceux-ci doivent être recommandés ou agréés par Siemens. Le fonctionnement correct et sûr des produits suppose un transport, un entreposage, une mise en place, un montage, une mise en service, une utilisation et une maintenance dans les règles de l'art. Il faut respecter les conditions d'environnement admissibles ainsi que les indications dans les documentations afférentes.

Marques de fabrique

Toutes les désignations repérées par ® sont des marques déposées de Siemens AG. Les autres désignations dans ce document peuvent être des marques dont l'utilisation par des tiers à leurs propres fins peut enfreindre les droits de leurs propriétaires respectifs.

Exclusion de responsabilité

Nous avons vérifié la conformité du contenu du présent document avec le matériel et le logiciel qui y sont décrits. Ne pouvant toutefois exclure toute divergence, nous ne pouvons pas nous porter garants de la conformité intégrale. Si l'usage de ce manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
ALLEMAGNE

A5E03638268-01
℗ 09/2011

Copyright © Siemens AG 2011.
Sous réserve de modifications techniques

Sommaire

1 Marques 5

2 Composants .NET pour COMOS 7

2.1 Composant .NET pour GUI 7
2.1.1 Afficher le contrôle pour l'application 7
2.1.2 Conditions requises pour un contrôle .NET dans COMOS 8
2.1.3 Application 11

2.2 Composants .NET pour les documents 11
2.2.1 Afficher le contrôle pour l'application 11
2.2.2 Conditions requises pour un contrôle .NET dans les documents 12
2.2.3 Application 13

3 Développement avec Visual Basic 6 dans COMOS 15

3.1 Recommandation 15

3.2 Conditions techniques requises 15
3.2.1 Enregistrer les composants COM 16
3.2.2 Enregistrer les Typelibs 16
3.2.3 Copier les fichiers Visual Basic dans le répertoire "Bin" 17

3.3 Déboguer les composants Visual Basic 18

3.4 Tester les composants Visual Basic 19

4 Configuration Side-by-Side à l'aide de manifestes 21

4.1 Manifestes 21

4.2 Création et mise à jour des manifestes pour les composants COM 22

4.3 Créer et mettre à jour des manifestes pour les assemblies .NET 25

4.4 Intégrer un manifeste à une assembly .NET 25

5 WinCC 27

6 ControlProperties 29

6.1 Accès à la propriété CtrlProperty via Lib.Sui 29

6.2 Traitement en bloc des ControlProperties via les requêtes d'objet 29

6.3 Liste des ControlProperties 30

6.4 Affectation des ControlProperties aux SuiControls 31

7 Créer et intégrer les symboles 3D 33

7.1 Introduction 33

7.2 Créer un projet VB 33

7.3 Créer un objet de base COMOS en 3D 34

7.4 Créer les procédures de dessin 35

7.5 Fonctions de dessin prédéfinies ... 37
7.5.1 Introduction ... 37
7.5.2 Blocs ... 37
7.5.3 Cylindres sur fond carré ... 38
7.5.4 Cylindre 1 (sol optionnel) ... 39
7.5.5 Cylindre 2 (sol et couvercle optionnels) ... 40
7.5.6 Segment de cylindre ... 41
7.5.7 Paroi de cylindre ... 42
7.5.8 Disque circulaire ... 43
7.5.9 Polygone1 ... 43
7.5.10 Polygone2 ... 44
7.5.11 Corps de rotation ... 44
7.5.12 Cylindre excentrique ... 45
7.5.13 Fond torosphérique ... 46
7.5.14 Pyramide ... 46
7.5.15 Cylindre ... 47
7.5.16 Sphère ... 48
7.5.17 Segment de sphère ... 48
7.5.18 Texte ... 49
7.5.19 Tore ... 49
7.5.20 Segment de tore ... 50
7.5.21 Bloc en mode fil de fer ... 51
7.5.22 Section de cylindre ... 52
7.5.23 Pyramide excentrique ... 53
7.5.24 Arc en tore (réduction possible) ... 54
7.5.25 Fonctions OpenGL ... 54
7.5.26 Générer des connecteurs ... 55
7.5.26.1 Exemples ... 56
7.5.27 Dimensions (grabs) ... 57
7.5.28 Points de saisie (points d'alignement) ... 58
7.5.29 Fonctions spéciales ... 58

8 Spécifications d'icônes ... 59

9 Autres composants ... 61

9.1 Report.ocx ... 61

9.2 UseSpecs.dll ... 61

9.3 ComosTGeneratorJobs ... 63
9.3.1 La classe NextString et la classe RunValues ... 63

Marques

1

Marques

Dépôt de la mention du droit de protection : COMOS®

Composants .NET pour COMOS 2

2.1 Composant .NET pour GUI

Les composants .NET peuvent s'utiliser aussi bien comme plug-ins que comme interfaces pour Device et CDevice. En d'autres termes : Désormais, il n'y a plus de différence entre les plug-ins et les onglets "Propriétés".

2.1.1 Afficher le contrôle pour l'application

Main.xml

Le contrôle utilisé doit être configuré pour l'application COMOS. Pour cela, il faut créer une entrée dans Main.xml (se référer au manuel "Démarrer COMOS").

Entrée `<Tool>`

Une entrée `<Tool>` doit exister au minimum pour chaque contrôle.

Exemple :

```
<ComosMainMenu>
  <Lib>
  <Tools>
  ...
    <Tool id="@NETWPFControl" type="Button">
    <Caption>
    <Text itx="WPFControl"/>
    </Caption>
    <Class type="NET" assembly="Comos.Test.WinFormsUC"
progid="Comos.Test.WinFormsUC.WPFTestControl"/>
    <Picture key="std_actionTry"/>
    <Parameters>
    <Parameter name="P1" value="WPF" />
    </Parameters>
  </Tool>
...
```

Contrôle au sein du menu COMOS

Pour afficher également le contrôle dans le menu COMOS, une entrée de référence `idref` dans le nœud `<Menu>` est nécessaire.

Exemple :

```
<ComosMainMenu>
  ...
  <Menu>
    <Toolbars>
    ...
    <Tool idref="Test">
    ...
    <Tool idref="@NETWPFControl"/>
```

Saisie de l'ID dans CDevice

Pour démarrer le contrôle par un double-clic sur un objet, la valeur de **<**`Tool id`**>** doit être saisie dans CDevice.

1. Dans les propriétés d'un objet de base, ouvrez l'onglet "Système".
2. Cliquez sur le bouton "Définir ControlTyp (ProgID) pour déterminer la fenêtre des propriétés".
3. Saisissez la valeur de **<**Tool id**>** dans le champ "ProgID de l'élément de commande".

   Auparavant, ce champ contenait le ProgID des anciens composants COM. Il faut saisir `<Tool id>` pour les composants .NET, étant donné que le ProgID est insuffisant. Outre le ProgID, d'autres informations comme le nom d'assembly au minimum, sont nécessaires. Ces informations supplémentaires sont prélevées à partir du `<Tool id>`.

2.1.2 Conditions requises pour un contrôle .NET dans COMOS

Condition requise minimale

`System.Windows.FrameworkElement`

En outre, `System.Windows.Controls.UserControl` est également autorisé.

Optionnellement, `System.Windows.Forms.UserControl` peut également être utilisé. Cela est toutefois déconseillé, car, en fin de compte, un contrôle Forms est hébergé dans `<WindowsFormsHost .../>`. Ainsi, l'utilisation directe de `System.Windows.Controls.UserControl` est plus efficace en termes de rapidité, de logique et de stabilité.

Communication entre .NET et les contrôles COMOS

Afin de pouvoir mieux communiquer avec le monde COMOS, le contrôle peut posséder deux propriétés `public : Workset` et `ItemObjects`. Le système les identifie par réflexion et les définit en conséquence :

```
  public Plt.IComosDWorkset Workset
  {
    get;
    set;
  }
  public object ItemObjects  // Ici, ItemObjects du type :
IComosDCollection ou IComosDBaseObejct ou VBA.Collection
  {
    get;
    set;
  }
```

La variante plus judicieuse consiste à implémenter l'interface `Comos.Controls.IComosControl` (`Comos.Controls.dll`).

```
namespace Comos.Controls
  {
  public interface IComosControl
  {
    Plt.IComosDWorkset Workset { get; set; }
    Plt.IComosDGeneralCollection Objects { get; set; }
    string Parameters { get; set; }
    IContainer ControlContainer { get; set; }
    void OnExecuted(ExecutedRoutedEventArgs e);
    void OnCanExecute(CanExecuteRoutedEventArgs e);
    void OnPreviewExecuted(ExecutedRoutedEventArgs e);
  }
```

`System.IDisposable` peut également être implémenté si nécessaire.

Le contrôle original est enregistré dans un conteneur qui contient le contrôle OkAbort et est chargé de l'administration Undostate. On peut accéder au conteneur pour activer et désactiver les boutons et éventuellement l'administration Undostate :

```
  public interface IContainer
  {
    object Child { get; set; }
    bool UndoStateAdministration { get; set; }  //default true
    void InitButton(StandardButtons button, string caption,
Visibility visibility);
    object UIFParent { get; }
    void Close();
  }
```

`OnExecuted, OnCanExecute, OnPreviewExecuted` s'appuient sur `RoutedUICommand` de WPF. Cela permet de commander les actions des boutons ainsi que les commandes de `CloseUIForm` et `ProjectOrWOChange`. Cependant, à la différence de WPF, `Executed` et `PreviewExecuted` peuvent être édités. Généralement, si l'on édite `PreviewExecuted` , aucun `Executed` n'est lancé avec WPF-RoutedUICommand, et cela indépendamment de `e.Handled`.

Dans le conteneur COMOS, les commandes sont traitées comme suit :

```
//. . .
Comos.Controls.IComosControl ctrl;
CanExecuteRoutedEventArgs e;
ExecutedRoutedEventArgs ee;
//. . .
ctrl.OnCanExecute(e);
if (e.CanExecute)
{
  ctrl.OnPreviewExecuted(ee);
  if (!ee.Handled)
  ctrl.OnExecuted(ee);
}
```

Exception pour le traitement de ProjectOrWOChange

Lors du traitement de `ProjectOrWOChange` , `OnExecuted` est appelé indépendamment de `ee.Handled`. A la différence des autres commandes, `ProjectOrWOChange` ne se rapporte pas à un seul contrôle, mais éventuellement à plusieurs. C'est pourquoi, une fois l'action lancée, elle ne doit pas être interrompue par un contrôle. Par contre, l'action n'a pas encore été démarrée pour `OnCanExecute` . Par conséquent, l'exécution peut encore être interrompue.

En principe, `ProjectOrWOChange` est judicieux uniquement pour les contrôles devant être conservés après un changement de projet. Par défaut, tous les contrôles sont désactivés lors d'un changement de projet. Si un contrôle doit être conservé après un changement de projet, l'entrée suivante est en outre nécessaire : `<Tool> <... autoClose="false">`.

Exemple :

```
<Tool id="@NET.UIFormsList" type="Button">
<Caption>
  <Text itx="~03416 Liste de fenêtres"/>
</Caption>
<Class type="NET" assembly="Comos.UIF"
progid="Comos.UIF.UIControls.UIFormsList"/>
<Picture key="std_windowList"/>
<Window type="Tab" area="Navigator" subarea="Detail"
autoClose="false" />
</Tool>
```

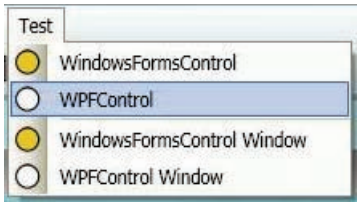
Démarrer les contrôles dans le code

Pour démarrer les contrôles dans le code (p. ex. pour le débogueur d'objets), procédez comme suit :

```
Set WS = a.Workset
ToolID = "@NETWPFControl"
' Démarrer en tant que "plug-in"
Set AppComd = WS.Globals.AppCommand
AppComd.Execute "ExecuteMenuByID", ToolID
' Démarrer en tant que "propriétés" :
Set App = WS.Globals.Application
App.ShowPropertiesOnMdiChild a, false, "", "CONTROLTYPE", ToolID
```

2.1.3 Application

Le résultat pourrait p. ex. se présenter comme suit :



Exemple

Menu COMOS, entrée "Aide > International > NET-Example_GUI"

2.2 Composants .NET pour les documents

Les composants .NET peuvent également être utilisés pour les documents. Pour cela, les composants .NET sont définis en tant que programmes au sein des types de documents. En double-cliquant sur un document, le composant correspondant s'ouvre.

2.2.1 Afficher le contrôle pour l'application

Main.xml

Le contrôle utilisé doit être configuré pour l'application COMOS. Pour cela, il faut créer une entrée dans Main.xml (se référer au manuel "Démarrer COMOS").

Entrée <Tool>

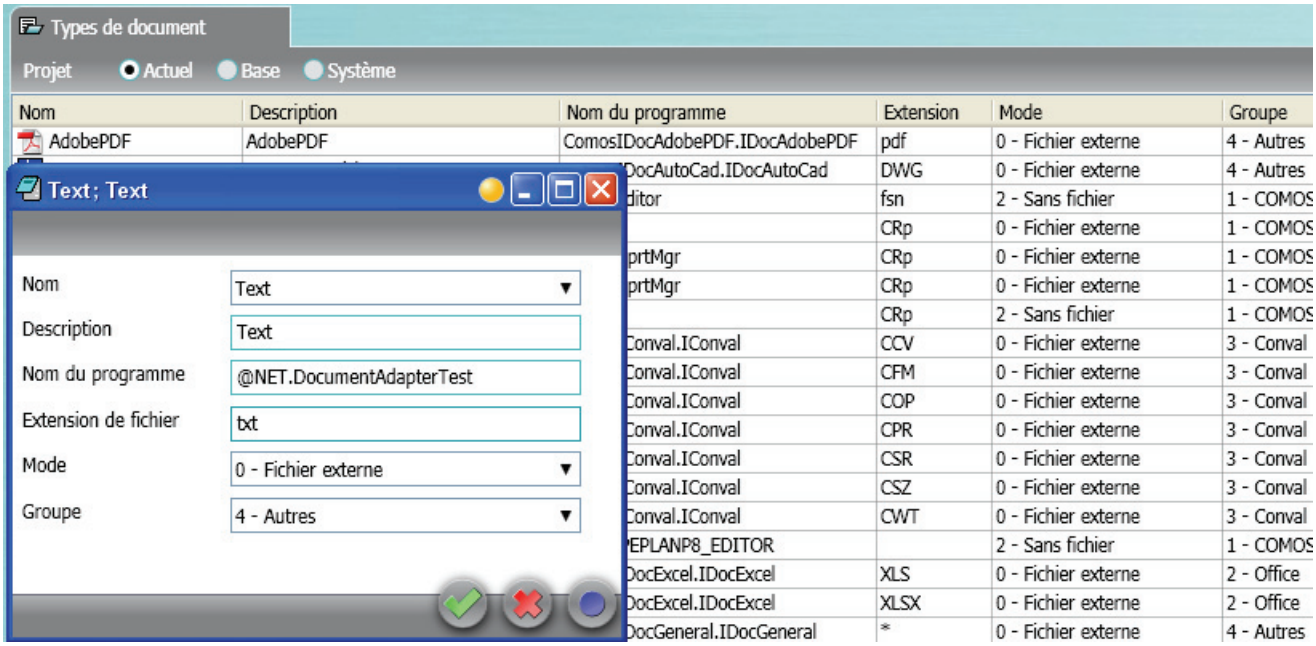
Une entrée <Tool> doit exister au minimum pour chaque contrôle.

Exemple :

```
<ComosMainMenu>
  <Lib>
  <Tools>
  …
  <Tool id="@NET.DocumentAdapterTest" type="Button">
  <Caption>
  <Text itx="DocumentAdapter Test"/>
  </Caption>
  <Class type="NET" assembly="Comos.Test.TextDocument"
progid="Comos.Test.TextDocument.TextDocControl"/>
  </Tool>
  …
  </Lib>
  …
</ComosMainMenu>
```

Créer un type de document

Il faut créer un type de document, dont le nom de programme est repris à partir de la valeur <`Tool id`> :



Auparavant, ce champ contenait le ProgID des anciens composants COM. Il faut saisir <`Tool id`> pour les composants .NET, étant donné que le ProgID est insuffisant. Outre le ProgID, d'autres informations comme le nom d'assembly au minimum, sont nécessaires. Ces informations supplémentaires sont extraites à partir du `<Tool id>`.

2.2.2 Conditions requises pour un contrôle .NET dans les documents

Condition requise minimale

`System.Windows.FrameworkElement`

En outre, `System.Windows.Controls.UserControl` est également autorisé.

Optionnellement, `System.Windows.Forms.UserControl` peut également être utilisé. Cela est toutefois déconseillé, car, en fin de compte, un contrôle Forms est hébergé dans `<WindowsFormsHost .../>`. Ainsi, l'utilisation directe de `System.Windows.Controls.UserControl` est plus efficace en termes de rapidité, de logique et de stabilité.

Informations complémentaires

Voir également Conditions requises pour un contrôle .NET dans COMOS (Page 8).

2.2.3 Application

Sur la base du nouveau type de document, un nouveau document peut être créé comme d'habitude. L'interface souhaitée s'affiche par un double-clic.

Exclusions

Pour des raisons techniques, seule l'ouverture du document est actuellement prise en charge.

Les fonctions d'impression, de révision etc. ne sont pas encore disponibles.

Exemple

Menu COMOS, entrée "Aide > International > NET-Example_Doc"

Développement avec Visual Basic 6 dans COMOS 3

3.1 Recommandation

Au sein de COMOS, les composants dans Visual Basic 6 sont pris en charge uniquement pour des raisons de compatibilité. Pour les nouveaux développements, utilisez la technique .NET.

Voir aussi

Composants .NET pour COMOS (Page 7)

3.2 Conditions techniques requises

En règle générale, lorsque vous installez COMOS, le programme fonctionne sans l'enregistrement des composants COM et utilise à la place la technologie Side-by-Side de Windows. Etant donné que Visual Basic ne prend pas en charge le technologie Side-by-Side, COMOS 9.0 doit être enregistré pour que vous puissiez effectuer des développements en Visual Basic sous COMOS 9.0.

Conditions requises

Les conditions requises suivantes sont requises pour le développement en Visual Basic :

- Les composants COM de COMOS doivent être enregistrés.
- Les Typelibs de COMOS doivent être enregistrés.
- Les fichiers Visual Basic doivent être disponibles dans le répertoire "Bin" de COMOS.

Voir aussi

Enregistrer les composants COM (Page 16)

Enregistrer les Typelibs (Page 16)

Copier les fichiers Visual Basic dans le répertoire "Bin" (Page 17)

3.2.1 Enregistrer les composants COM

Condition requise

Pour que l'enregistrement fonctionne, utilisez la version la version 1,0,0,1 du fichier "ComosRegister.dll" et enregistrez-la. Le fichier est disponible dans le répertoire suivant :

"<Répertoire d'installation COMOS>\updatecenter"

Marche à suivre

Pour enregistrer les composants COM, procédez comme suit :

1. Ouvrez le dossier "Bin" dans le répertoire d'installation de COMOS.
2. Pour enregistrer tous les composants COM dans les répertoires "OCX" et "ComosX.ocx", exécutez le fichier "ComosReg.exe".

Résultat

Les composants COM sont enregistrés.

3.2.2 Enregistrer les Typelibs

Condition requise

Le fichier de script "RegTLB.bat" avec lequel vous enregistrez les Typelibs de COMOS, requiert le mappage du lecteur R. Le fichier script est disponible dans le répertoire suivant :

"<Répertoire d'installation COMOS>\Current\Bin\DeveloperReg"

Marche à suivre

Pour effectuer le mappage du lecteur R, procédez comme suit :

1. Appelez la ligne de commande.
2. Dans la ligne de commande, exécutez la ligne suivante :

```
Subst R: <Répertoire d'installation COMOS>
```

Résultat

Le lecteur R est mappé.

Autre solution

Pour effectuer le mappage d'un autre lecteur, procédez comme suit :

1. Ouvrez le fichier de script "RegTLB.bat".
2. Dans ce fichier, adaptez toutes les références au répertoire d'installation de COMOS.

Marche à suivre

Pour enregistrer les Typelibs de COMOS, procédez comme suit :

1. Appelez le répertoire suivant : "<Répertoire d'installation COMOS>\Current\Bin\DeveloperReg"
2. Pour enregistrer les Typelibs, exécutez le fichier de script "RegTLB.bat" dans ce répertoire.

Résultat

Les Typelibs de COMOS sont enregistrés.

3.2.3 Copier les fichiers Visual Basic dans le répertoire "Bin"

Condition requise

Vous utilisez Visual Basic 6.0 SP6.

Marche à suivre

Pour copier les fichiers Visual Basic dans le répertoire "Bin", procédez comme suit :

1. Appelez le répertoire d'installation de Visual Basic.
2. Copiez les fichiers suivants dans "<Répertoire d'installation COMOS>\Current\Bin" :
   - "C2.exe"
   - "CVPACK.exe"
   - "LINK.exe"
   - "MSDIS110.dll"
   - "MSPDB60.dll"
   - "VB6.exe"
   - "VB6.olb"
   - "VB6DEBUG.dll"

- "VB6EXT.olb"
- "VB6IDE.dll"
- "VBA6.dll"
- "VBAEXE6.lib"

Les fichiers "VB6.exe.config" et "VB6.exe.Manifest" doivent également être disponibles. En règle générale, ces deux fichiers sont installés lors de l'installation de COMOS.

Remarque

N'ouvrez pas le fichier VBP par un double-clic dans Explorer. Cela a pour conséquence que le fichier erroné "VB6.exe" est utilisé ou que la fonction définie pour le double-clic est modifiée.

3.3 Déboguer les composants Visual Basic

Condition requise

Tous les composants sont mis à jour pour la version COMOS 9.0.

Recommandation

Pour éviter les problèmes liés aux droits d'accès, utilisez votre ordinateur local pour le développement.

Marche à suivre

Pour déboguer les composants Visual Basic, procédez comme suit :

1. Dans "<Répertoire d'installation COMOS>\Current\Bin", exécutez le fichier "VB6.exe".
2. Ouvrez le projet "ComosDebug", disponible dans le répertoire suivant :

   "<Répertoire d'installation COMOS>\Current\Bin\ComosDebug"
3. Ajoutez le projet VB6 modifié.
4. Démarrez le projet.

Résultat

COMOS s'ouvre et vous pouvez déboguer en Visual Basic.

Remarque

Mode débogage

En mode de débogage, les événements des composants .NET ne sont pas systématiquement pris en compte dans leur totalité.

3.4 Tester les composants Visual Basic

Vous pouvez tester les composants nouvellement compilés en les copiant dans le répertoire d'installation COMOS.

Configuration Side-by-Side à l'aide de manifestes 4

Si vous utilisez une version COMOS antérieure à 9.0, vous devez vous servir de la méthode conventionnelle et enregistrer les composants COM, p. ex. via "RegSvr32".

Autre solution à partir de Windows XP SP2 :

A partir de Windows XP SP2, vous pouvez optionnellement copier un composant COM à l'aide de la configuration Side-by-Side et l'utiliser sans enregistrement. Copiez le composant COM dans le même répertoire que celui contenant le fichier EXE.

4.1 Manifestes

Pour la configuration Side-by-Side, vous devez ajouter un manifeste au fichier EXE.

Définition

Le manifeste est un fichier XML contenant des indications sur les conditions requises par l'application EXE pour l'exécution. Ce fichier contient les indications pour l'enregistrement COM. Il s'agit généralement des informations figurant dans le registre Windows. C'est pourquoi, l'application peut être exécutée même si les composants COM ne sont pas enregistrés.

Enregistrer les manifestes

Vous pouvez enregistrer les manifestes en tant que fichiers externes dans des fichiers EXE en utilisant le nom suivant :

<Nom de l'application EXE>.manifest

Optionnellement, vous pouvez intégrer les manifestes en tant que ressources au sein des ressources de programme.

Assemblies COM

S'agissant des assemblies COM dans COMOS, les manifestes sont enregistrés en tant que fichiers externes. Le dossier "Bin" contient un fichier de manifestes avec les types de COM pour le fichier EXE correspondant.

Restrictions

Les restrictions suivantes s'appliquent dans le cadre de la configuration Side-by-Side :

- Etant donné que dans COMOS 9.0, les programmes COM ne sont plus enregistrés dans le registre Windows, ils sont dotés d'un fichier de manifeste permettant de configurer uniquement localement les types de COM via le fichier EXE correspondant. C'est pourquoi, dans le dossier "Bin", il existe un fichier exe.manifest pour chaque fichier EXE.
- Aucune assembly COM livrée par COMOS (fichiers OCX et DLL) n'est désormais enregistrée par défaut dans le registre Windows par l'intermédiaire de "RegSvr32". En revanche, toutes les informations sur les assemblies sont enregistrées dans le fichier "ComosOcx.manifest".
- Un manifeste permet d'accéder aux références et répertoires situés en dessous uniquement à partir du niveau de répertoire correspondant. Il s'agit généralement du répertoire "Bin".

  Le répertoire "OCX" se trouve dans le dossier "Bin".
- Le procédé Side-by-Side n'est pas adapté aux ordinateurs de développement mais uniquement aux applications disponibles sur les ordinateurs des utilisateurs. Sur l'ordinateur de développement, l'enregistrement est toujours nécessaire.

Programmes externes

Les règles suivantes s'appliquent aux programmes externes accédant à COMOS via COM selon le procédé Side-by-Side :

- Tous les programmes externes, y compris COM et .NET, doivent être enregistrés dans le dossier "Bin" de l'installation locale de COMOS. Copiez, p. ex., le programme "CustomComosExtension.exe" dans le répertoire suivant : "<Chemin d'installation de COMOS >\Current\Bin"
- Tous les fichiers .NET-DLL visibles à partir de COM (techniquement : COM visible), doivent être enregistrés dans le dossier "Bin".
- Créez un fichier manifeste approprié afin que les objets COMOS puissent être accédés à partir du fichier externe "CustomComosExtension.exe" via interop COM.

4.2 Création et mise à jour des manifestes pour les composants COM

Vue d'ensemble

Si vous souhaitez intégrer un composant COM à un fichier manifeste ou si un fichier manifeste est adapté lors d'une modification d'interface, vous pouvez pour cela utiliser le programme "Comos.ManifestMaker.exe".

Exemple

Vous utilisez un fichier COM-DLL, p. ex. un fichier DLL dans Visual Basic, appelé "CustomComosExtension.dll". Dans la ligne de commande ou dans un fichier script, saisissez la ligne suivante et exécutez-la :

"C:\ManifestMaker\Comos.ManifestMaker.exe" -type:COM -filename:C:\Custom\MyCustomComosExtension.dll -output:C:\Custom

- Le premier paramètre est le type de l'assembly (`COM` ou `DOTNET`).
- Le deuxième paramètre est le nom de l'assembly.
- Le troisième paramètre est optionnel et détermine l'endroit où le fichier manifeste créé sera enregistré ainsi que son nom.

Résultat

Après la création du fichier manifeste, le résultat se présente comme suit, p. ex. :

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
 <file name="MyCustomComosExtension.dll">
 <typelib tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" version="1.8"
 helpdir="" />
 <comClass description="MyCustomComosExtension"
 progid="CustomComosExtension.HelperClass1"
 clsid="{69DAA847-D481-446E-967B-2DE3EFBBB672}"
 tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" />
 <comClass description="MyCustomComosExtension"
 progid="CustomComosExtension.HelperClass2"
 clsid="{9F40C2F3-96D4-40cf-83B9-6D85DDE8E0E9}"
 tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" />
 </file>
</assembly>
```

Nœud `File`

Le nœud `File` contient des informations sur le composant. Pour l'ajouter au fichier "ComosCustomizeOcx.manifest", procédez comme suit :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity
      name="ComosCustomizeOcx"
      version="1.0.0.0"
      type="win32" />

  <!-- ************************************************************ -->
  <!-- **              ComosCustomizeOcx.manifest                ** -->
  <!-- **                                                        ** -->
  <!-- **   References to additional Assemblies or OCX Components ** -->
  <!-- **                                                        ** -->
  <!-- **   please insert here manifest information for own      ** -->
  <!-- **   components, that exist only at specific customer     ** -->
  <!-- **   installation                                         ** -->
  <!-- ************************************************************ -->

  <file name="Custom\MyCustomComosExtension.dll">
    <typelib tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" version="1.8"
        helpdir="" />
    <comClass description="MyCustomComosExtension"
        progid="CustomComosExtension.HelperClass1"
        clsid="{69DAA847-D481-446E-967B-2DE3EFBBB672}"
        tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" />
    <comClass description="MyCustomComosExtension"
        progid="CustomComosExtension.HelperClass2"
        clsid="{9F40C2F3-96D4-40cf-83B9-6D85DDE8E0E9}"
        tlbid="{B7FFF897-DA01-4b88-AD41-F75CF213FE9C}" />
  </file>

</assembly>
```

Dans cet exemple, le fichier "MyCustomComosExtension.dll" est enregistré dans le sous-répertoire "Custom" du dossier "Bin". C'est pourquoi, l'attribut `Name` est adapté en conséquence par le nœud `File` .

4.3 Créer et mettre à jour des manifestes pour les assemblies .NET

Marche à suivre

Pour créer et mettre à jour des manifestes pour les assemblies .NET, procédez comme suit :

1. Utilisez l'outil Comos.ManifestMaker.exe via la ligne de commande.
2. Appelez la ligne de commande.
3. Dans la ligne de commande, saisissez la ligne suivante :

   "<Installationsverzeichnis con COMOS>\PTV001_Vega\Bin\COMOS.ManifestMaker.exe" -type:DOTNET -filename:C:\Custom\ComosCustom.dll –output:C:\Custom
4. Exécutez vos saisies.

Résultat

Le fichier manifeste généré est enregistré dans le répertoire "Custom".

Remarque

Veuillez noter qu'un fichier manifeste créé avec "mt.exe" se trouve toujours dans le même répertoire que le composant attribué au sein du fichier manifeste.

Recommandation

Dans le cas d'une assembly .NET (application ou bibliothèque), intégrez le manifeste au fichier binaire final.

Voir aussi

Intégrer un manifeste à une assembly .NET (Page 25)

4.4 Intégrer un manifeste à une assembly .NET

Condition requise

Pour intégrer un manifeste à une assembly .NET, utilisez les paramètres `autoembed` et `mtpath`.

Marche à suivre

Pour intégrer un manifeste à une assembly .NET, procédez comme suit :

Remarque

Vous devez utiliser le programme "mt.exe" de Microsoft pour l'intégration des fichiers manifestes aux assemblies .NET. Ce programme est inclus dans Windows SDK à partir de Windows Vista ou dans Visual Studio .NET SDK à partir de Visual Studio .NET 2005. Les SDK sont téléchargeables gratuitement sur la page d'accueil de Microsoft.

1. Ouvrez l'invite de saisie avec accès au répertoire comprenant le fichier "mt.exe".
2. Exécutez la commande suivante pour l'assembly :

```
"C:\Comos\PTV001_Vega\Bin\Comos.ManifestMaker.exe" -type:DOTNET
-filename:C:\Custom\ComosCustom.dll –autoembed –mtpath:C:\Tools
```

Résultat

Un manifeste de type "RT_MANIFEST" avec ID "1" est intégré à l'assembly en tant que ressource.

WinCC 5

Exemple

Dans l'exemple suivant, la chaîne de caractères est dimensionnée dans un premier temps (`char ASString[250]`). Ensuite, une variable est préparée dans laquelle les `SystemUID` de COMOS peuvent être enregistrés (`char SystemUID[20]`).

Paramètres spécifiques à WinCC

Vous trouvez ici les paramètres spécifiques à WinCC : le bloc `if-else` est utilisé pour les recherches d'erreurs.

Remarque

Les UID sont mentionnés à titre d'exemple uniquement. Votre base de données contient d'autres UID.

L'UID figurant dans la ligne `if-` est différente de celle des deux lignes `else` suivantes (sinon, le bloc `if-else` ne serait pas judicieux). Dans le cas de données réelles, des milliers, voire des dizaines de milliers d'objets (p. ex. les signaux) seront probablement à gérer. Les boucles `if-else` ne sont alors plus judicieuses et doivent être remplacées par des détecteurs d'erreurs avancés.

Ensuite, la chaîne qui a été dimensionnée dans la première ligne est composée ("`strcpy`").

Paramètres autorisés et commutateurs

Référez-vous au manuel "Démarrer COMOS", mot-clé "Paramètres".

Les commutateurs qui y sont décrits peuvent être ajoutés à la chaîne de caractères ("`strcat`").

Exemple de programme

La chaîne enregistrée dans "ASString" est alors transmise à la commande "ProgramExecute". ProgramExecute repère d'abord l'appel de XIF au sein de la chaîne et l'exécute ensuite avec les paramètres suivants :

```
{
char ASString[250] ;
char SystemUID[20];

if (GetTagBit("U2") == 1)
   strcpy(  SystemUID, "/ID: C1D45BO8U7  " );
else if (GetTagBit("Erreur") == 1)
   strcpy(  SystemUID, "/ID: A1D9VBO5U7  " );
else
   strcpy(  SystemUID, "/ID: A1D9VBO5U7  " );

strcpy( ASString, "C:\\Programme\\COMOS\\Bin\\XIF.exe  " );
strcat( ASString, "/DB: ACCESS ");
strcat( ASString, "/DN: C:\\COMOS_DB\\COMOSPT.MDB " );
strcat( ASString, "/PR: CeBIT " );
strcat( ASString, SystemUID );
strcat( ASString, "/PA: DETAIL" );

ProgramExecute( ASString );
}
```

ControlProperties

6

6.1 Accès à la propriété CtrlProperty via Lib.Sui

Les ControlProperties peuvent être définies par script au moyen d'une constante. Optionnellement, la chaîne des ControlProperties peut être décomposée puis recomposée.

Exemple d'accès en écriture

```
a.workset.lib.sui.CtrlProperty (27,a) = 0...'27 = Left
a.workset.lib.sui.CtrlProperty (28,a) = 0...'28 = Top
a.workset.lib.sui.CtrlProperty (29,a) = 20...'29 = Right
a.workset.lib.sui.CtrlProperty (30,a) = 5...'30 = Bottom
```

Exemple d'accès en lecture

```
cp=0...'0 = LabelSize
Output a.workset.lib.sui.CtrlProperty (cp,a)
```

6.2 Traitement en bloc des ControlProperties via les requêtes d'objet

Marche à suivre

- Créer une requête standard pour les attributs.
- Charger une requête à partir d'un fichier XML (p. ex. : QueryControlProperties.xml; voir le répertoire d'installation de COMOS, sous-dossier Aide\ Français\ Autres documentations\ Exemples de ControlProperties XML.zip).
- Ajouter des colonnes supplémentaires pour les propriétés souhaitées :
  - Copiez la colonne sous Options > Edition des colonnes,
  - renommez-la,
  - passez à numérique, le cas échéant,
  - modifiez la constante cp de la propriété pour lui donner la valeur souhaitée sous les propriétés des scripts "Calcul de valeurs" et "Objet par valeur" (cf. tableau).
- Les colonnes des ControlProperties sont éditables.
- Pour transmettre des ControlProperties à partir de CSpecification vers BackPointer, vous devez également charger une requête à partir d'un fichier XML
  (p. ex. : ControlPropertiesToBackPointer.xml; voir le répertoire d'installation de COMOS, sous-dossier Aide\Français\Autres documentations\ Exemples de ControlProperties XML.zip).
  Modifiez les constantes cpX (v. tableau) des propriétés dans le script du bloc d'actions.

6.3 Liste des ControlProperties

ControlProperties

Constante	Nom	Description	Type de données	Valeurs possibles
0	LabelSize	Largeur de la description	Double	
1	NameSize	Largeur du nom	Double	
2	UnitControlSize	Largeur de l'unité	Double	
3	ShowLabelNameDescr	Description/Afficher le nom	Entier	0 = description uniquement 1 = nom et description
4	ControlType	Type de champ de saisie	Entier	1 = Champ de texte 2 = Combobox éditable 3 = Combobox
5	UnitControlType	Unité fixe	Entier	2 = Combobox 3 = unité fixe
6	ProductRequestLbSize	Largeur des données de produit	Double	
7	ScrollBarActiv	Barre de défilement active	Entier	0= Non 1= Oui
8	ButtonVisible	Bouton visible	Entier	0= Non 1= Oui
9	LabelAlignment	Alignement du texte	Double	0 = Gauche 1 = Droite
10	LabelFontName	Nom de la police	Chaîne	
11	LabelFontSize	Taille de la police	Double	
12	LabelFontBold	Police en gras	Boolean	
13	LabelFontItalic	Police en italique	Boolean	
14	LabelFontUnderLine	Police soulignée	Boolean	
15	LabelForeColor	Couleur de la police	Long	
16	ProductRequestKeyInput	Données de produit éditables localement	Entier	0 = Non éditable 1 = Toujours 2 = Si l'objet de conception est une "requête"
17	SpecNames (en lecture uniquement !)	Préfixes de nom de l'attribut	Chaîne	Modifiable à partir de l'interface uniquement
18	MaxCount (en lecture uniquement !)	Nombre maximal de jeux de données	Entier	Modifiable à partir de l'interface uniquement
19	Non utilisable		(Désactivé)	
20	Non utilisable		(Désactivé)	
21	ShowLinkObjectName	Affichage	Entier	0 = Description de l'objet lié 1 = Nom de l'objet lié 2 = FullName de l'objet lié 3 = DisplayValue de l'attribut

Constante	Nom	Description	Type de données	Valeurs possibles
22	KeyInput	Mode d'affichage : non éditable	Entier	0 = Editable 1 = Non éditable
23	QueryFullName	FullName système de la requête d'objet	Chaîne	
24	FilterStdTable	Filtrer les listes de sélection	Entier	0 = Désactivé 1 = Côté données de base 2 = Toujours
25	BtnClickAllowed	Bouton exécutable	Long	0 = <gestion normale des droits> 1 = Toujours (indépendamment des droits)
26	BorderType	Type de cadre	Entier	Type de cadre 1 = Normal 2 = Simple 3 = En relief 4 = Abaissé
27	Left			
28	Top			
29	Right			
30	Bottom			
31	CaptionVisible	Titre de cadre visible	Boolean	
32	Mode	Boutons	Variant	En tant que sommes binaires : 1 = Sélection de fichier 2 = Editer la feuille Excel 4 = Calculer la feuille Excel 8 = Attribution
33	ExecuteValue	Exécuter la requête immédiatement	Entier	0= Non 1= Oui
34	LabelFontStrikethrough	Police barrée	Boolean	
35	Buttonstyle	Mode d'affichage : compact	Entier	0= Tous les boutons 1= Compact

6.4 Affectation des ControlProperties aux SuiControls

Légende de la liste d'affectation (colonnes)

A = Edit	E = Excel	I = Repeater	M = Query
B = Border	F = Check	J = Memo	N = FOpen
C = Button	G = Image	K = Memo_rtf	O = Link
D = Label	H = Signature	L = Date	P = List

Liste d'affectation

	Nom	A	B	C	C	E	F	G	H	I	J	K	L	M	N	O	P
0	LabelSize	+				+			+		+		+		+	+	
1	NameSize	+									+					+	
2	UnitControlSize	+															
3	ShowLabelNameDescr	+									+					+	
4	ControlType	+															
5	UnitControlType	+															
6	ProductRequestLbSize	+															
7	ScrollBarActiv										+						
8	ButtonVisible										+						
9	LabelAlignment						+										
10	LabelFontName	+	+	+	+		+		+		+		+		+	+	
11	LabelFontSize	+	+	+	+		+		+		+		+		+	+	
12	LabelFontBold	+	+	+	+		+		+		+		+		+	+	
13	LabelFontItalic	+	+	+	+		+		+		+		+		+	+	
14	LabelFontUnderLine	+	+	+	+		+		+		+		+		+	+	
15	LabelForeColor	+	+		+		+		+		+		+		+	+	
16	ProductRequestKeyInput	+															
17	SpecNames (read only)									+							
18	MaxCount (read only)									+							
19	(Désactivé)																
20	(Désactivé)																
21	ShowLinkObjectName															+	
22	KeyInput															+	
23	QueryFullName													+			
24	FilterStdTable	+															
25	BtnClickAllowed			+													
26	BorderType		+														
27	Left	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
28	Top	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
29	Right	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
30	Bottom	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+
31	CaptionVisible		+														
32	Mode					+											
33	ExecuteValue													+			
34	LabelFontStrikethrough	+	+	+	+		+		+		+		+		+	+	
35	Buttonstyle															+	

Créer et intégrer les symboles 3D

7

7.1 Introduction

Ce chapitre décrit comment créer un symbole 3D à l'aide de Visual Basic, l'enregistrer dans une dll et le rendre disponible pour Viper dans COMOS.

7.2 Créer un projet VB

Préparation

Créez d'abord un nouveau projet VB.

Déclarez le module et l'interface

```
Option Explicit

Const ModuleName = "XXX Objects"

Private Geo As Comos3D.I3DGeoObject

Public Property Set GeometrieInterface(ByVal vNewValue As I3DGeoObject)

    Set Geo = vNewValue

End Property

Public Property Let GeometrieInterface(ByVal vNewValue As I3DGeoObject)

    Set GeometrieInterface = vNewValue

End Property
```

A présent, les procédures de dessin 3D peuvent être créées ainsi que, parallèlement, les objets de base COMOS correspondants.

A l'issue de ce processus, vous devez créer le fichier dll du projet et l'enregistrer dans le répertoire d'installation COMOS "\<COMOS> \ OCX \ GL". Le nouveau fichier est ensuite enregistré dans ce répertoire (ouvrir avec WINNT\system32\regsvr32.exe).

7.3 Créer un objet de base COMOS en 3D

Introduction

Tout objet de base placé dans COMOS Viper, doit posséder un onglet appelé "GD". Cet onglet contient tous les principaux attributs pour l'affichage 3D.

Caractéristiques applicables à tous les attributs pertinents pour l'affichage 3D : Onglet "Lien". "Mode 3D" : activé.

Les attributs déjà existants se trouvent dans la base de données livrée, accessible via le chemin "@Viper > @Y > @3D > @PP > 01 Onglet GD". Les attributs déjà existants de cette base de données devraient être réutilisés pour les nouveaux objets. Les attributs portant le nom "VC" + nombre doivent obligatoirement être repris à partir de ce répertoire. Ils décrivent les connecteurs auxquels COMOS accède directement.

Attributs pour chaque composant 3D

- "VGEO Type géo" :
  Contient l'appel de la procédure de dessin sous la forme :
  (Nom du ficher cls, sans "Objects") + "." + nom de la fonction de dessin
- Attributs nécessaires, spécifiques à chaque composant de tuyau :
  - "VPCL Classe de tuyau" :
    Définit la classe de tuyau d'un objet.
  - "VFCD Code de fonction" :
    Attribue l'objet à un groupe d'objets similaires. Sert au mappage des classes de tuyau et à l'identification d'un type d'objet au sein du code.

Attributs optionnels

- "LAYER Calque" :
  Définit un groupe d'objets affichés et masqués en commun.
- "VCOL Couleur" :
  Définit la couleur d'un objet.
- "VR01 Code AR" :
  Définit l'état d'un objet lors du routage. Si cet attribut est inexistant, le routeur tient compte du composant.
- "VSUI Attributs éditables" :
  Contient le nom de tous les attributs, séparés par une virgule, qui s'affichent sur l'interface 3D lorsque l'objet correspondant est sélectionné. Ici, vous n'avez pas besoin de paramétrer le mode 3D étant donné que cet attribut peut également être utilisé hors mode Viper.

7.4 Créer les procédures de dessin

Définition d'une procédure avec un nom significatif (p. ex. "box" comme nom de procédure pour un bloc). En outre, chaque fonction de dessin devrait être dotée d'un ErrorHandler :

```
Public Property Set GeometrieInterface(ByVal vNewValue As I3DGeoObject)

    Set Geo = vNewValue

End Property

Public Property Let GeometrieInterface(ByVal vNewValue As I3DGeoObject)

    Set GeometrieInterface = vNewValue

End Property

Public Sub Box()
    On Error GoTo ComosStdErrorHandler

' CSErrorHandler begin - please do not modify or remove this line
Exit Sub
ComosStdErrorHandler:
StdErrorHandler Err, ModuleName, "Sub Box"
End Sub
```

Déclaration des variables et importation des valeurs d'attribut à partir de l'onglet GD à l'aide de la commande `Geo.Value` ("Nom d'attribut").

```
Public Property Let GeometrieInterface(ByVal vNewValue As I3DGeoObject)

    Set GeometrieInterface = vNewValue

End Property

Public Sub Box()
    On Error GoTo ComosStdErrorHandler

    Dim length As Double
    Dim width As Double
    Dim height As Double

    length = Geo.Value("V002")
    width = Geo.Value("V050")
    height = Geo.Value("V012")

' CSErrorHandler begin - please do not modify or remove this line
Exit Sub
ComosStdErrorHandler:
StdErrorHandler Err, ModuleName, "Sub Box"
End Sub
```

Dans COMOS, la fonction de dessin est appelée à plusieurs reprises. A chaque appel, une ou plusieurs parties du symbole sont dessinées ou mises à jour. C'est pourquoi, `Geo.GeometrieMode` permet de différencier les états de l'actuel appel de procédure. Comme indiqué ci-dessous, il existe sept états différents dont chacun est défini dans une requête "If". Lors de la procédure, il n'est pas nécessaire de définir les GeometrieModes superflus.

```
Public Sub Box()
    On Error GoTo ComosStdErrorHandler

    Dim length As Double
    Dim width As Double
    Dim height As Double

    length = Geo.Value("V002")  ''SY
    width = Geo.Value("V050")   ''SX
    height = Geo.Value("V012")  ''SZ

    If Geo.GeometrieMode.GenerateConnectors Or Geo.GeometrieMode.GenerateGeometrie
    End If

    If Geo.GeometrieMode.GenerateGrabs Then
    End If

    If Geo.GeometrieMode.GenerateSnapPoints Then
    End If

    If Geo.GeometrieMode.GenerateCenterLines Then
    End If

    If Geo.GeometrieMode.GenerateOutlines Then
    End If

    If Geo.GeometrieMode.SurfaceRendering Then
    End If
```

A présent, vous pouvez dessiner la forme du symbole dans la boucle "If" avec l'expression `Geo.GeometrieMode.GenerateGeometrie.`.

7.5 Fonctions de dessin prédéfinies

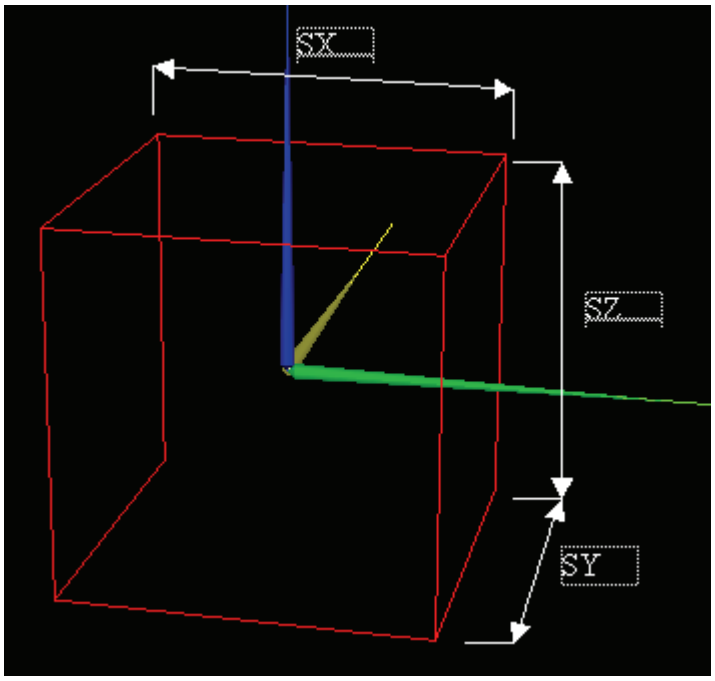
7.5.1 Introduction

Les axes de coordonnées tracés indiquent le point zéro et l'orientation du composant (vert : axe X, jaune : axe Y, bleu : axe Z).

7.5.2 Blocs

Fonction

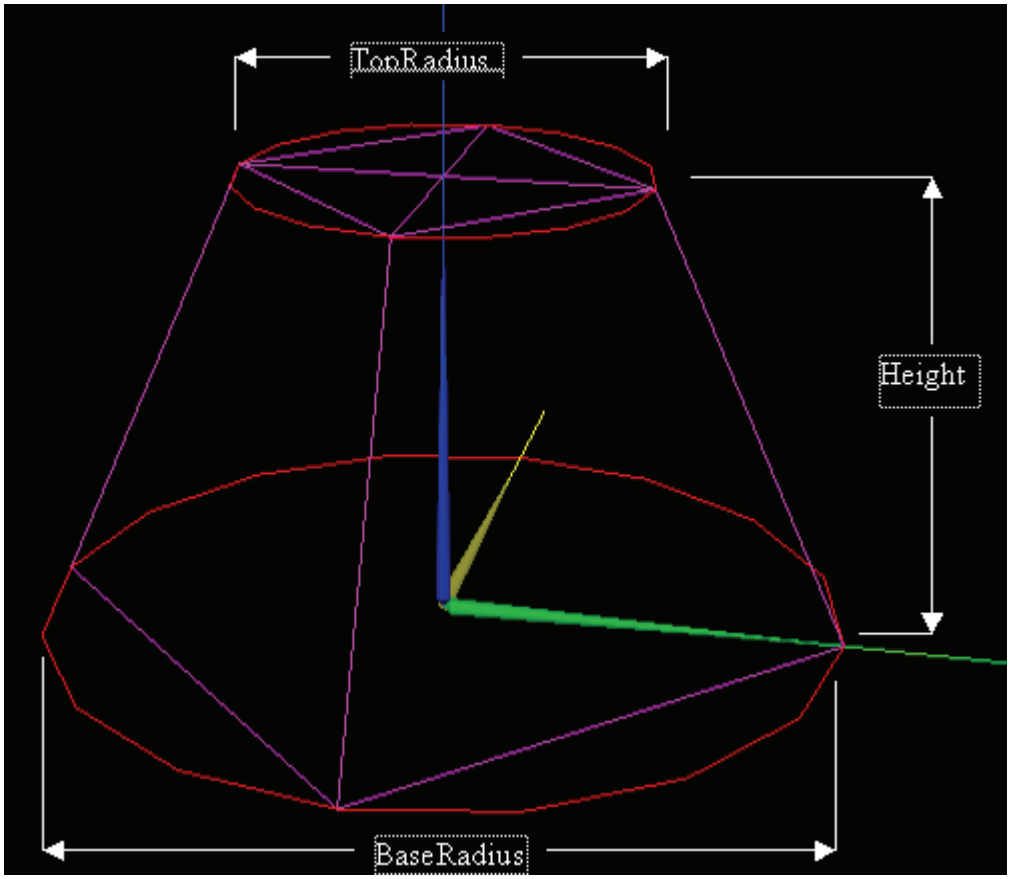
```
Geo.Box
SX As Double,
SY As Double,
SZ As Double
```



7.5.3 Cylindres sur fond carré

Fonction

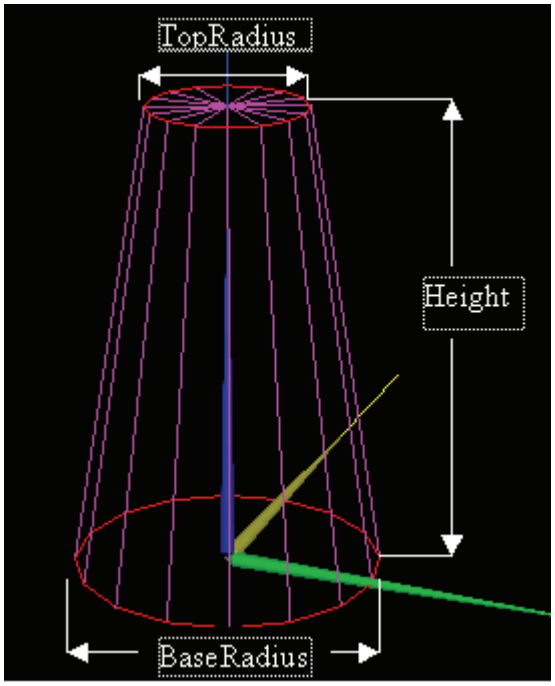
```
Geo.BoxCylinder
BaseRadius As Double,
TopRadius As Double,
Height As Double,
[Top As Boolean=vrai],
[Bottom As Boolean=vrai]
```



7.5.4 Cylindre 1 (sol optionnel)

Fonction

```
Geo.BoxCylinder
BaseRadius As Double,
TopRadius As Double,
Height As Double,
[Top As Boolean=vrai],
[Bottom As Boolean=vrai]
```

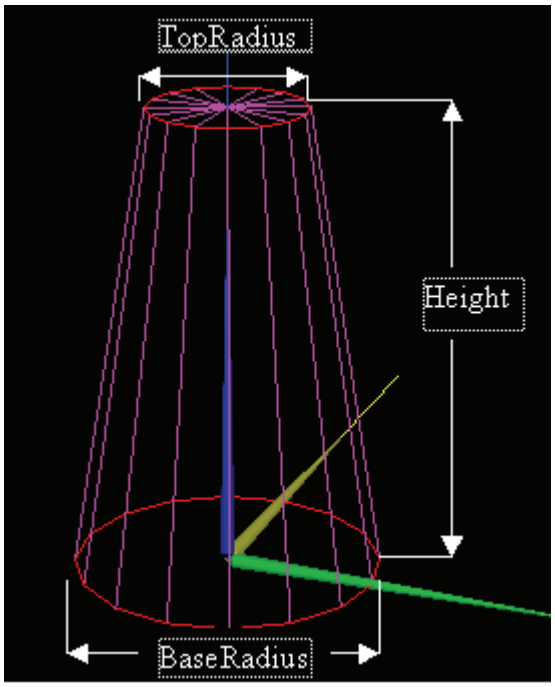


7.5.5 Cylindre 2 (sol et couvercle optionnels)

Fonction

```
Geo.Cylinder
BaseRadius As Double,
TopRadius As Double,
Height As Double,
[Top As Boolean=vrai],
[Bottom As Boolean=vrai]
```

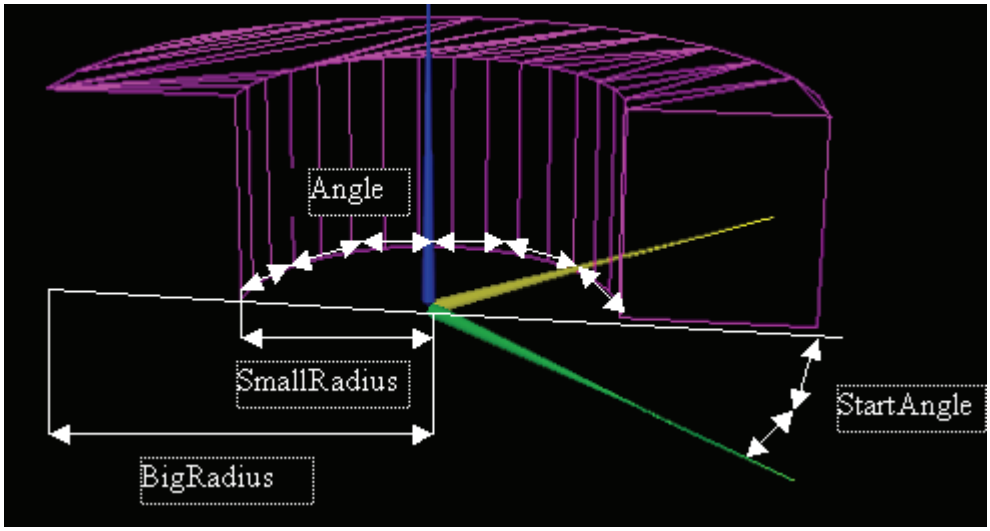
`Top` et `Bottom` déterminent la représentation du sol et du couvercle.



7.5.6 Segment de cylindre

Fonction

```
Geo.CylinderSegment
BigRadius As Double,
SmallRadius As Double,
Height As Double,
StartAngle As Double,
Angle As Double
```

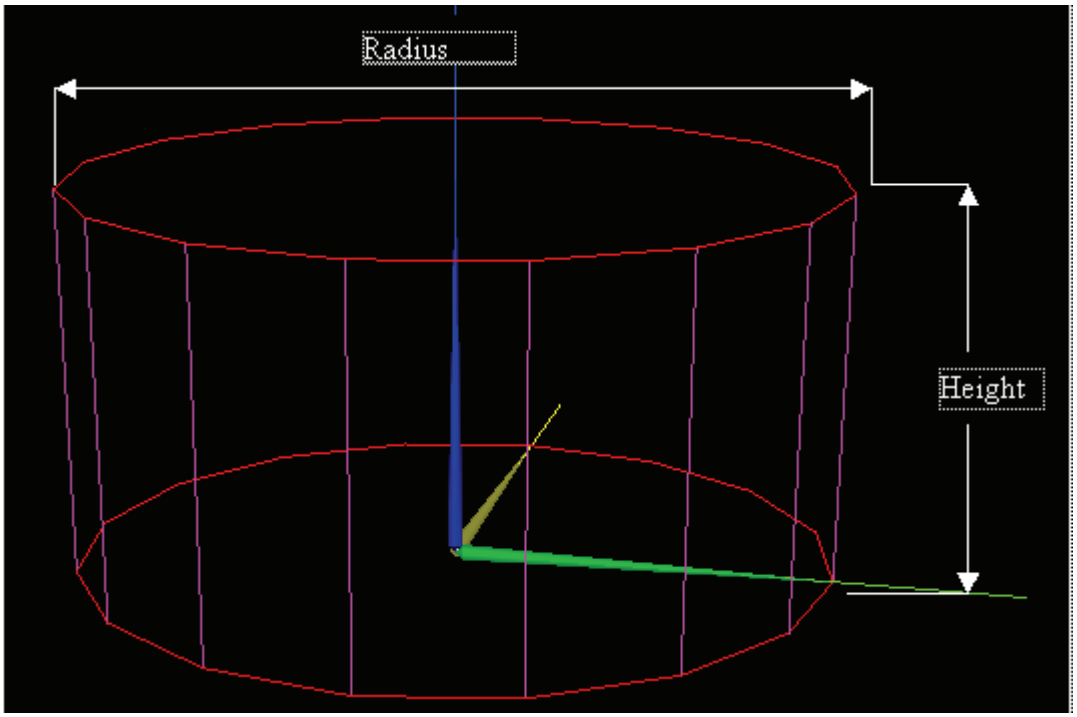


7.5.7 Paroi de cylindre

Fonction

```
Geo.CylinderSide
Outside As Boolean,
Radius As Double,
Height As Double
```

`Outside` détermine la visibilité de la surface extérieure.

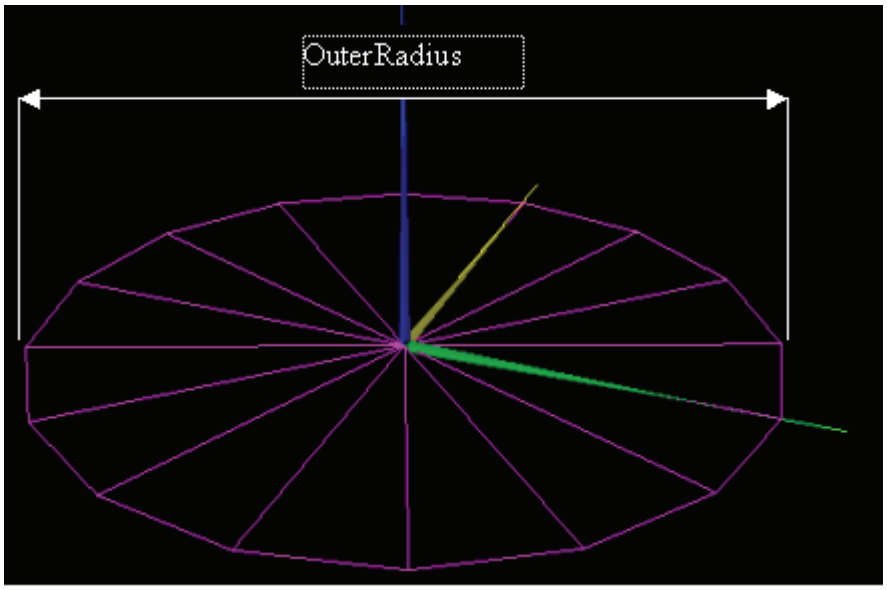


7.5.8 Disque circulaire

Fonction

```
Geo.CylinderSide
Outside As Boolean,
Radius As Double,
Height As Double
```

`Outside` détermine la visibilité de la surface extérieure.



7.5.9 Polygone1

Fonction

```
Geo.Extrusion
NumPoints As Long,
Contur() As POINT3D,
Height As Double,
[Side_Lines As Boolean=vrai]
```

La valeur `Height` détermine l'étendue du polygone selon l'axe Z.

7.5.10 Polygone2

Fonction

```
Geo.Polygon
NumPoints As Long,
Contur() As POINT3D,
XNormal As Double,
YNormal As Double,
ZNormal As Double,
[Reverse As Boolean]
```

`XNormal`, `YNormal` et `ZNormal` déterminent la surface à représenter. Définition du vecteur de normales

7.5.11 Corps de rotation

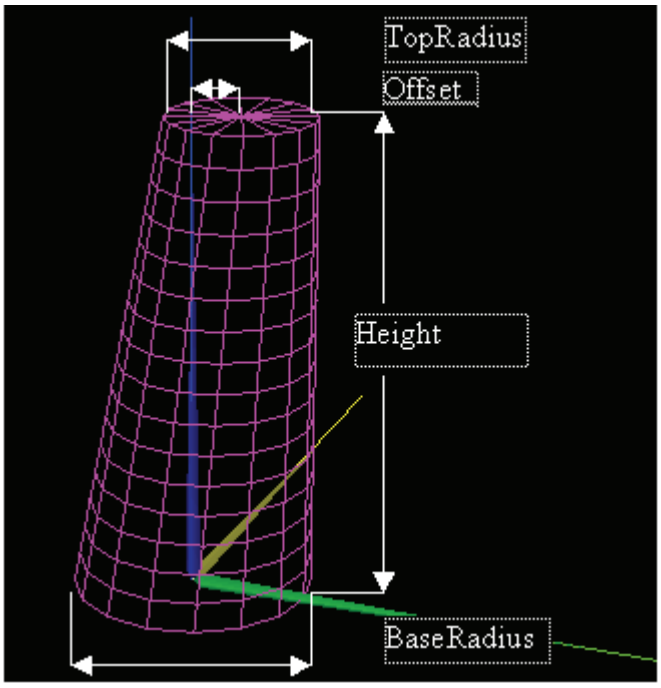
Fonction

```
Geo.Rotation
NumPoints As Long,
Contur() As POINT3D,
Angle As Double
```

7.5.12 Cylindre excentrique

Fonction

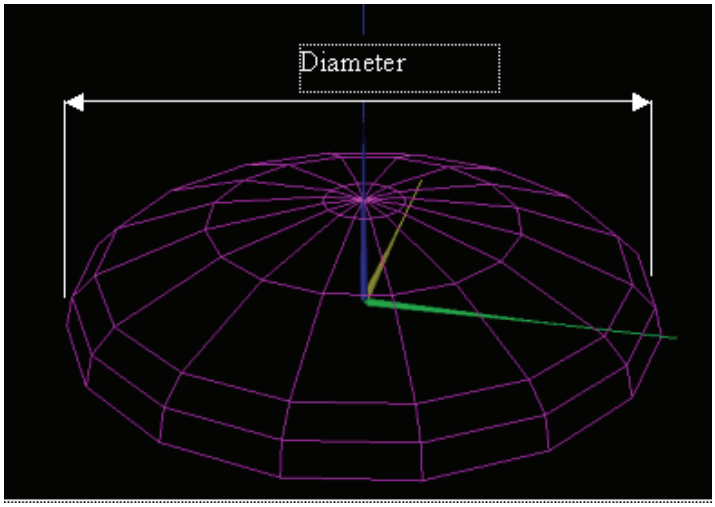
```
Geo.ExzCylinder
BaseRadius As Double,
TopRadius As Double,
Height As Double,
Offset As Double,
[Top As Boolean=vrai],
[Bottom As Boolean=vrai]
```



7.5.13 Fond torosphérique

Fonction

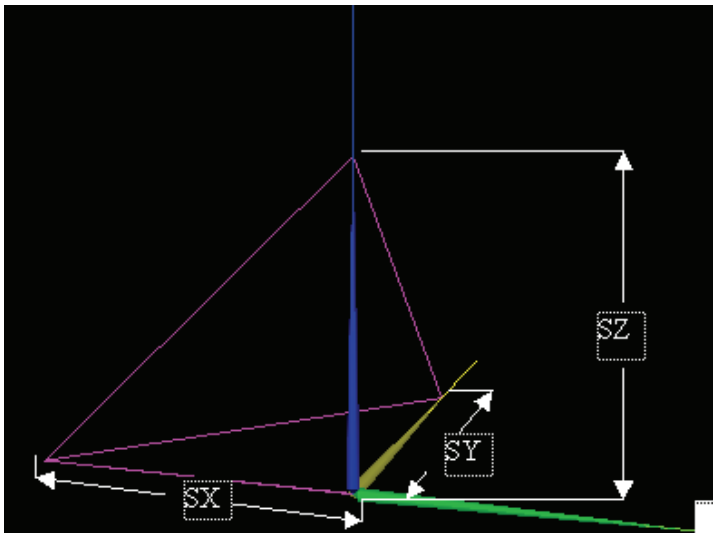
```
Geo.KPBoden
Diameter As Double
```



7.5.14 Pyramide

Fonction

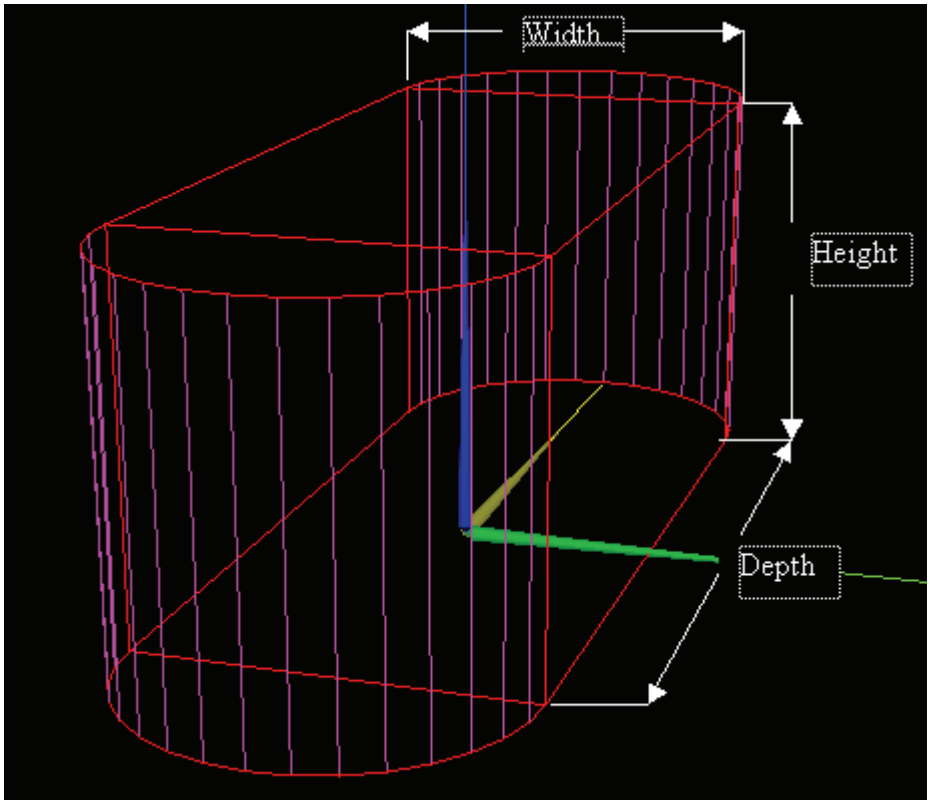
```
Geo.Pyramid
SX As Double,
SY As Double,
SZ As Double
```



7.5.15 Cylindre

Fonction

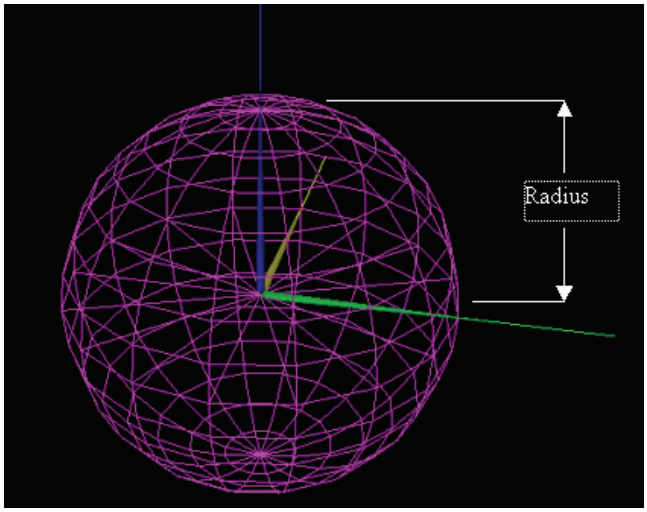
```
Geo.SheetWithRoundedCorner
Depth as Double,
Width As Double,
Height As Double
```



7.5.16 Sphère

Fonction

```
Geo.Sphere
Radius As Double
```

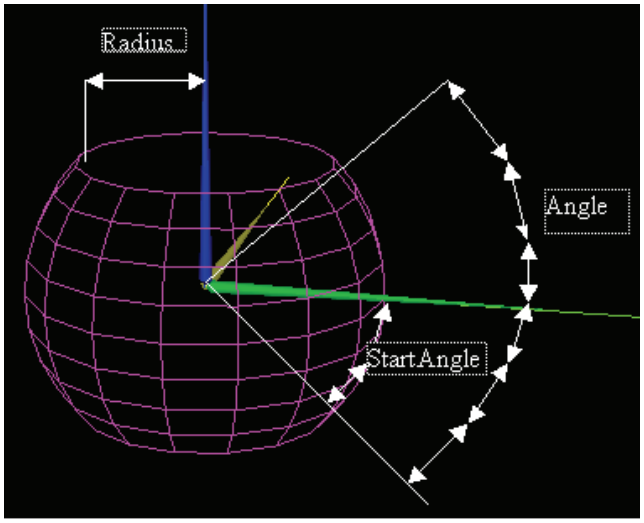


7.5.17 Segment de sphère

Fonction

```
Geo.SphereSegment
Radius As Double,
StartAngle As Double,
Angle As Double
```

(-90 < StartAngle < 90; Angle + StartAngel <=180)



7.5.18 Texte

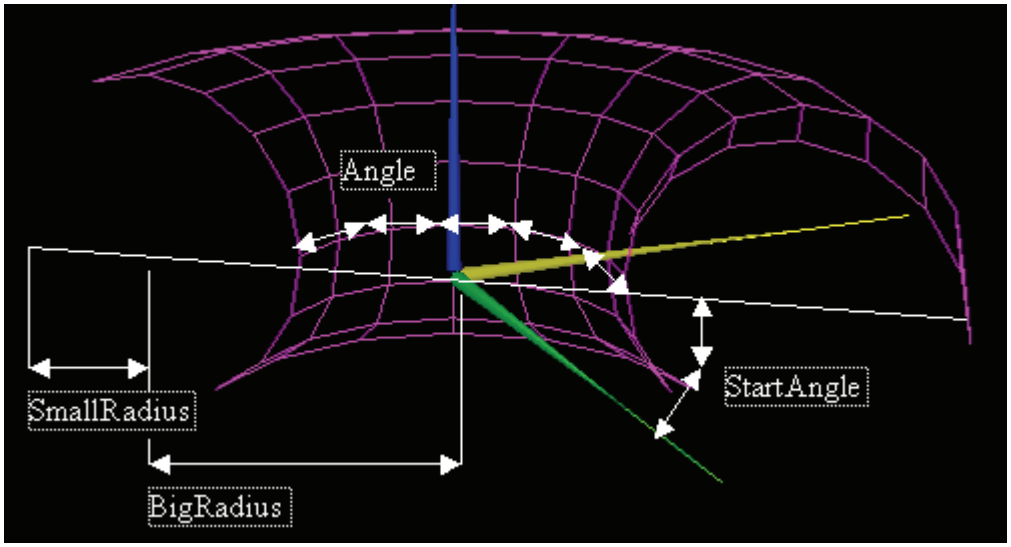
Fonction

```
Geo.Text
sText As String,
Height As Double
```

7.5.19 Tore

Fonction

```
Geo.TorusArc
SmallRadius As Double,
BigRadius As Double,
[StartAngle As Double],
[Angle As Double=360]
```

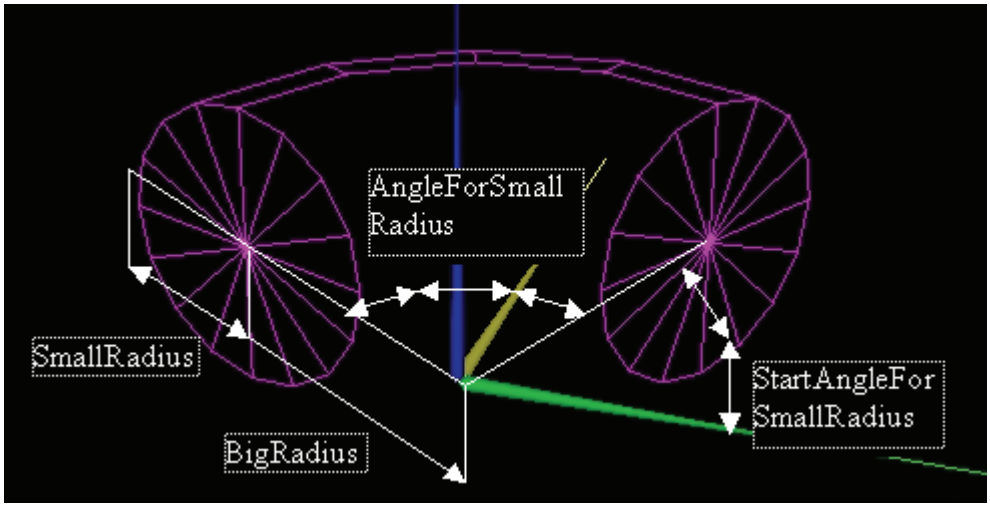


7.5.20 Segment de tore

Fonction

```
Geo.TorusArcSegment
SmallRadius As Double,
BigRadius As Double,
[StartAngleForBigRadius As Double],
[AngleForBigRadius As Double],
[StartAngleForSmallRadius As Double],
[AngleForSmallRadius As Double=360]
```

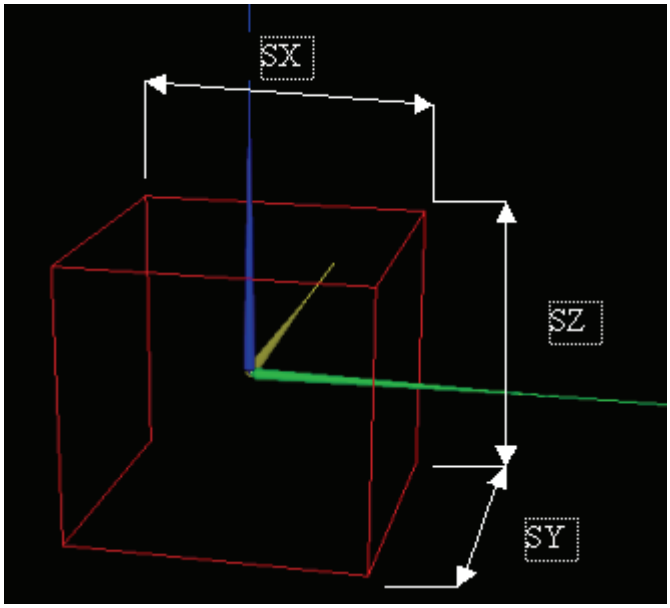
Le dessin de la surface extérieure est déterminé par `StartAngleForBigRadius` et `AngleForBigRadius` :



7.5.21 Bloc en mode fil de fer

Fonction

```
Geo.WireBox
SX As Double,
SY As Double,
SZ As Double
```

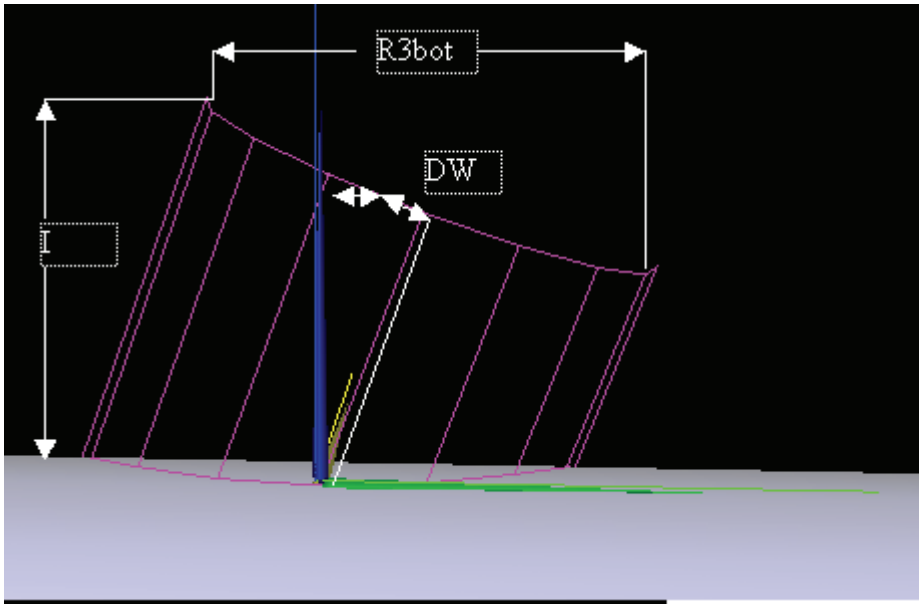


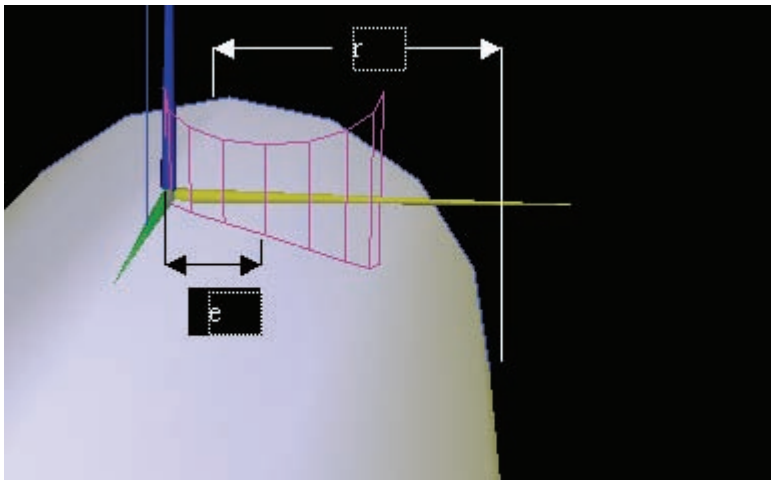
7.5.22 Section de cylindre

Fonction

```
Geo.CylinderCylinderIntersection
r3top As Double,
r3bot As Double,
r As Double,
I As Double,
[DW As Double],
[e As Double],
[PHI As Double]
```

`r3top` et `PHI` ne sont pas utilisés actuellement.



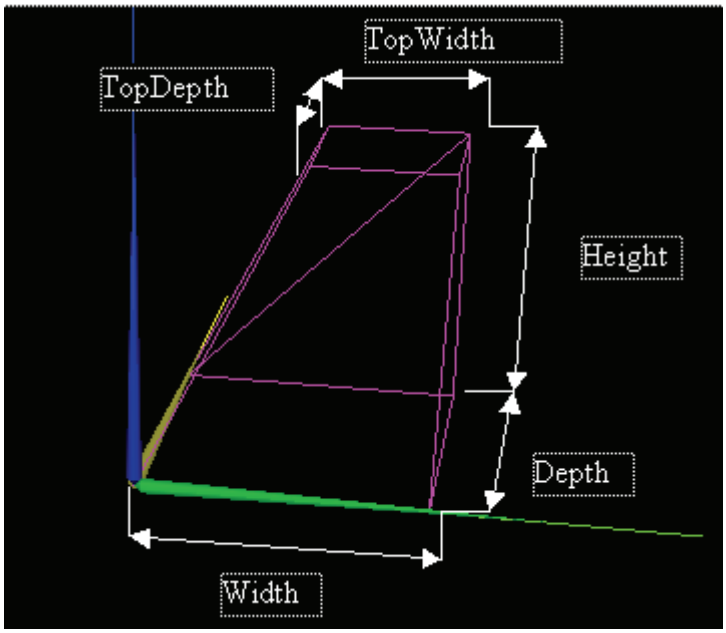


7.5.23 Pyramide excentrique

Fonction

```
Geo.ExzPyramide
Width As Double,
Depth As Double,
Height AS Double,
[TopWidth As Double],
[TopDepth As Double],
[OffsetX As Double],
[OffsetY As Double]
```

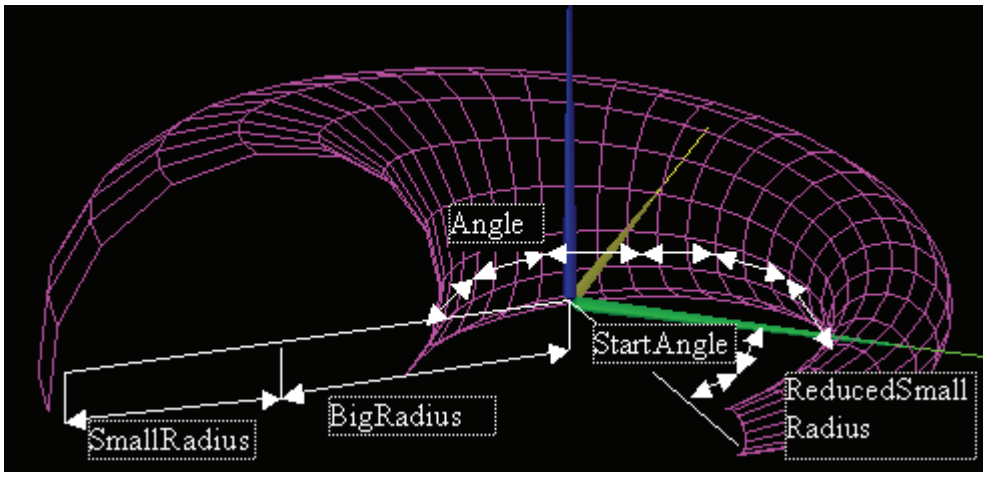
`OffsetX` et `OffsetY` décrivent le désaxement de la surface du couvercle par rapport au sol selon les axes X et Y.



7.5.24 Arc en tore (réduction possible)

Fonction

```
Geo.TorusArc1
SmallRadius As Double,
BigRadius As Double,
[StartAngle As Double],
[Angle As Double = 360],
[ReducedSmallRadius As Double = -1],
[NumberOfSegments As Long = -1]
```



7.5.25 Fonctions OpenGL

Naviguer dans l'espace 3D

Des fonctions dérivées d'OpenGL sont en outre disponibles pour la navigation dans l'espace 3D :

- Rotation autour d'un axe quelconque

  ```
  Geo.Rotate
  AngleDegree As Double,
  X AS Double,
  Y As Double,
  Z As Double
  ```

- Rotation autour d'un certain axe de l'objet

  – `Geo.RotateX`
    `AngleDegree As Double`

  – `Geo.RotateY`
    `AngleDegree As Double`

  – `Geo.RotateZ`
    `AngleDegree As Double`

- Décalage par rapport à la position actuelle
  ```
  Geo.Move
  X As Double,
  Y As Double,
  Z As Double
  ```
- Mémoriser la position actuelle
  ```
  Geo.PushMatrix
  ```
- Retourner à la position mémorisée
  ```
  Geo.PopMatrix
  ```
- Retourner au point zéro absolu
  ```
  Geo.LoadIdentity
  ```
- Prendre la position de la matrice indiquée
  ```
  Geo.LoadMatrix m As MATRIX3D
  ```

7.5.26 Générer des connecteurs

Connecter un symbole à des connecteurs (général)

Afin de connecter le symbole nouvellement créé à un autre symbole, il faut le doter d'un connecteur. Cela s'effectue de manière analogue au dessin de la forme de base, dans la boucle "If" mentionnée ci-avant. Déplacez-vous dans l'espace jusqu'à la position à laquelle le connecteur est placé sur le symbole, vérifiez l'orientation (voir les commandes ci-avant) et dessinez le connecteur au moyen de l'appel de fonction :

```
Geo.GeoConnector
(Index As Long, [ConnectorClass As
eConnectorClass=ConnectorClass_General] As I3Dconnector)
```

L'index du connecteur doit être > 0. Dans ce contexte, on distingue six classes de connecteurs avec différentes propriétés :

- `ConnectorClass_Elo` (pour les objets d'électrotechnique)
- `ConnectorClass_General` (paramètre par défaut)
- `ConnectorClass_Pipe` (pour les pièces de tuyaux, les robinetteries, etc.)
- `ConnectorClass_Steel` (pour les objets de construction en acier)
- `ConnectorClass_Undef`
- `ConnectorClass_VesselNozzle` (pour les réservoirs et les colonnes)

Remarques particulières concernant ConnectorClass_Pipe

A noter : Si vous créez des connecteurs du type ConnectorClass_Pipe, vous devez veiller à ce qu'ils contiennent des informations sur leur largeur et leur pression nominale ainsi que sur le type de connexion :

- VC11, VC12 et VC13 doivent être disponibles dans tout objet avec des connecteurs du type ConnectorClass_Pipe sous l'onglet "GD".

Vous pouvez en outre créer (optionnellement) les attributs suivants pour définir le nouveau connecteur :

- "VC" + index du connecteur + "1" pour la largeur nominale
- "VC" + index du connecteur + "2" pour la pression nominale
- "VC" + index du connecteur + "3" pour le type de connexion

Si l'un de ces attributs n'est pas disponible, le programme tente d'accéder progressivement à des attributs avec des index de connecteurs inférieurs.

Si un connecteur de ce type est doté d'un "raccord à bride", la surface de bride est dessinée automatiquement. Pour cela, il faut sélectionner la pression nominale du connecteur.

Exemple :

L'attribut "VC11" est disponible sous l'onglet "GD". A présent, si l'on crée un connecteur du type ConnectorClass_Pipe avec l'index 4, le programme procède comme suit :

Etant donné que "VC41" n'est pas disponible, le programme tente d'accéder d'abord à "VC31", puis à "VC21". Comme ces deux attributs sont également indisponibles, la valeur est finalement reprise à partir de l'attribut "VC11".

7.5.26.1 Exemples

Exemple 1

La description de symbole d'un bloc avec des connecteurs non orientés à chaque angle pourrait se présenter comme suit :

```
length = Geo.Value("V002")   ''SY
width = Geo.Value("V050")    ''SX
height = Geo.Value("V012")   ''SZ

If Geo.GeometrieMode.GenerateConnectors Or Geo.GeometrieMode.GenerateGeometrie
    Geo.Box width, length, height ''Zeichnen eines Quaders

    Geo.Move -width / 2, -length / 2, -height / 2 ''vom Mittelpunkt der Box an
    Geo.GeoConnector 1
    Geo.Move 0, 0, height ''Ecke vorne, oben, links
    Geo.GeoConnector 2
    Geo.Move 0, length, 0 ''Ecke hinten, oben, links
    Geo.GeoConnector 3
    Geo.Move 0, 0, -height ''Ecke hinten, unten, links
    Geo.GeoConnector 4
    Geo.Move width, 0, 0 ''Ecke hinten, unten, rechts
    Geo.GeoConnector 5
    Geo.Move 0, 0, height ''Ecke hinten, oben, rechts
    Geo.GeoConnector 6
    Geo.Move 0, -length, 0 ''Ecke vorne, oben, rechts
    Geo.GeoConnector 7
    Geo.Move 0, 0, -height '' Ecke vorne, unten, rechts
    Geo.GeoConnector 8
End If

If Geo.GeometrieMode.GenerateGrabs Then
End If
```

Exemple 2

La description de symbole d'un bloc avec des connecteurs orientés à chaque angle pourrait se présenter comme suit :

```
If Geo.GeometrieMode.GenerateConnectors Or Geo.GeometrieMode.GenerateGeometrie
    Geo.Box width, length, height ''Zeichnen eines Quaders

    Geo.Move -width / 2, -length / 2, -height / 2 ''vom Mittelpunkt der Box an
    Geo.PushMatrix
        Geo.RotateY 180
        Geo.RotateZ -90
        Geo.GeoConnector 1
    Geo.PopMatrix
    Geo.Move 0, 0, height ''Ecke vorne, oben, links
    Geo.PushMatrix
        Geo.RotateZ 180
        Geo.GeoConnector 2
    Geo.PopMatrix
    Geo.Move 0, length, 0 ''Ecke hinten, oben, links
    Geo.PushMatrix
        Geo.RotateZ 90
        Geo.GeoConnector 3
    Geo.PopMatrix
    Geo.Move 0, 0, -height ''Ecke hinten, unten, links
    Geo.PushMatrix
        Geo.RotateY 180
        Geo.GeoConnector 4
    Geo.PopMatrix
    Geo.Move width, 0, 0 ''Ecke hinten, unten, rechts
    Geo.PushMatrix
        Geo.RotateY 180
```

7.5.27 Dimensions (grabs)

Ensuite, vous pouvez définir une dimension à un attribut affecté en mode `Geo.GeometrieMode.GenerateGrabs`. Cela permet de modifier en ligne un objet dans le plan 3D.

```
        Geo.PopMatrix

        Geo.PopMatrix
    End If

    If Geo.GeometrieMode.GenerateGrabs Then
        Dim g As I3DGrab

        Geo.PushMatrix
        Geo.Move -width / 2, -length / 2, 0 ''Navigieren zum Beginn der Bemaßung

        Geo.Grab "V050" ''Erzeugen der Bemaßung
        Geo.Move width, 0, 0 ''Navigieren zum Ende der Bemaßung
        Geo.RotateZ 180 ''Rotation, Richtung des Grabs
        Set g = Geo.Grab("V050")
        g.MoveOrigin = False
        g.ShowDimension = False ''Soll ursprüngliche Bemaßungslänge angezeigt werden?
        g.Symmetrical = False
        Geo.PopMatrix
    End If

    If Geo.GeometrieMode.GenerateSnapPoints Then
    End If
```

7.5.28 Points de saisie (points d'alignement)

Le mode `Geo.GeometrieMode.GenerateSnapPoints` permet de définir des points de saisie supplémentaires.

7.5.29 Fonctions spéciales

Les fonctions spéciales suivantes sont disponibles :

- `Geo.GeometrieMode.GenerateOutlines`
- `Geo.GeometrieMode.GenerateCenterLines`

Geo.GeometrieMode.GenerateCenterLines

Le mode `Geo.GeometrieMode.GenerateOutlines` permet de définir des lignes extérieures supplémentaires. Dans ce mode, l'implémentation est à éviter car elle peut créer des erreurs fondamentales.

Geo.GeometrieMode.GenerateCenterLines

Le mode `Geo.GeometrieMode.GenerateCenterLines` permet de définir des lignes supplémentaires. Dans ce mode, l'implémentation est à éviter car elle peut créer des erreurs fondamentales.

Spécifications d'icônes

8

Spécifications techniques des icônes

- 72 dpi
- RGB 8-bit
- 16x16
  Seules les icônes d'une taille de 16x16 sont officiellement prises en charge. L'affichage correct de tous les autres formats au sein de COMOS n'est pas garanti systématiquement.
- Mise à l'échelle :
  La règle suivante s'applique à partir de COMOS 8.2 : les rapports hauteur-largeur ne sont pas conservés automatiquement. Le mode de représentation est adapté localement au sein de COMOS.
- Arrière-plan : transparent
  Pour des raisons d'interopérabilité (avec VB), la PicturesLibrary est obligée de marquer une couleur comme transparente lors de la conversion entre les deux mondes (COM et .NET). Pour cela, la PicturesLibrary se réfère au standard Microsoft et sélectionne la couleur du pixel inférieur gauche de l'image (sauf si celui-ci est noir).
  Par conséquent, lors de l'affichage d'une image nouvellement créée ou existante dont la seule couleur est celle du pixel inférieur gauche, l'image devient invisible ou cette couleur n'apparaît pas sur l'image.

Autres composants 9

9.1 Report.ocx

Document : Fonction

```
CreateDWG ( x1 as Double, y1 as Double, x2 as Double, y2 as Double ) as Object
```

`CreateDWG` crée un caractère générique (= rectangle) pour un fichier DWG/DXF du rapport. Les paramètres sont les coordonnées du caractère générique. La valeur renvoyée par la classe ReportDWG est un objet. La propriété FileName de cet objet détermine le fichier AutoCAD à afficher.

CreateDWG est toujours utilisable en cas d'accès à `ComosReportDocument` (à savoir la classe "Document" du rapport .ocx) Dans les modèles de rapport, c'est le cas de tous les objets placés qui possèdent un onglet Script au sein de leur propriétés.

Script

```
Set MyDwg = ReportDocument.CreateDWG(x1, y1, x2, y2)
MyDWG.FileName = "xyz.dwg"
```

VisualBasic

Intégrer un rapport .ocx

```
Dim MyDwg as ReportLIB.ReportDWG

Set MyDwg = ReportDocument.CreateDWG(x1, y1, x2, y2)

MyDWG.FileName = "xyz.dwg"
```

9.2 UseSpecs.dll

Fournit toutes les utilisations au moyen de la fonction `GetUseSpecifications` . Ainsi, cette fonction est l'équivalent en matière de scripts de l'onglet "Utilisation".

GetUseSpecifications

```
Function GetUseSpecifications(ByVal Spec As IComosDSpecification,
ByVal ObjType As ObjectType, ByVal DocType As DocumentType, ByVal
ProjType As ProjectType, ByVal Inherit As Boolean) As
IComosDCollection
Spec:
ObjType:
DocType:
ProjType:
Inherit:
```

Option ObjectType ("Evaluation de :")

0: aucun

1: objets de base

2: objets de conception

3: tous

Option DocumentType ("Documents concernés")

0: aucun

1: modèles

2: conception

3: tous

Option ProjectType ("Projet")

0: actuel

1: tous

Exemple 1

L'exemple suivant fournit toutes les utilisations de l'attribut "`CurrentSpec`". Seuls les objets de base et les modèles de documents sont alors pris en compte dans le projet actuel.

En outre, l'évaluation s'effectue avec des objets hérités (paramètre `True`).

```
Set UseSpesDLL = CreateObject("ComosUseSpecs.UseSpecs")
  If Not UseSpesDLL  Is Nothing Then
  Set SpecCol = UseSpesDLL.GetUseSpecifications (CurrentSpec, 1, 1,
0, True)
End If
```

Exemple 2

Sortie d'un attribut dans le débogueur d'objets

```
'CurrentSpec est alors, p. ex., l'objet déplacé par Glisser&Déposer
vers le champ de saisie "A"  =>
UseSpesDLL.GetUseSpecifications (A, 1, 1, ....
Output "SpecCol.Count = " + cstr(SpecCol.Count)
For i = 1 To SpecCol.Count
  Output SpecCol.item(i).systemfullname
Next
```

9.3 ComosTGeneratorJobs

9.3.1 La classe NextString et la classe RunValues

Le CObject contient soit le masque, soit @MASK:NomEnregDonnéesDansTableau

Class NextString

Fonction

- AddItem
  `AddItem ( ByVal ItemStr as String ) as Integer`
  Permet d'ajouter un élément de la chaîne.
- MaskValid
  `MaskValid ( ) as Boolean`
  Permet de tester si le masque est correct.
- Test
  `Test ( ByVal CurrentValue as String ) as Boolean`
  Permet de tester la chaîne transmise du masque actuel et indique si la chaîne est valide.
- GetNext
  `GetNext ( ) as String`
  Fournit la chaîne calculée suivante et n'ajoute pas d'article.
- RegExpPattern
  `RegExpPattern ( ByVal ComosMask as String ) as String`
- TestPrefix
  `TestPrefix ( ByVal CurValue as String ) as Boolean`

Propriété

- DefaultValue
  `DefaultValue ( ) as String`
- Mask
  `Mask ( ) as String`
  Masque de texte COMOS, donc modèle de la nouvelle chaîne.
- ScreenMessages
  `ScreenMassages ( ) as Boolean`

Sub

- Clear
  `Clear ( )`
  Supprime tous les éléments de la chaîne.

Class RunValues

Fonction

- Mask

  ```
  Mask ( ByVal TextGenerator as IComosDTextGenerator ) as String
  ```

- DefaultValue

  ```
  DefaultValue ( ByVal TextGenerator as IComosDTextGenerator ) as String
  ```

Propriété

- ScreenMessages

  ```
  ScreenMassages ( ) as Boolean
  ```